Logic in AI

ITI0210, lecture 6 (2021)

Review

Search:

- **steps to solution**
- **optimization**
- games

Bold areas are useful in implementing logic algorithms

Knowledge and Reasoning

What’s the point of using logic?

Knowledge and Reasoning

Knowledge

Wizards wear a wizard hat
Gandalf wears a wizard hat

Reasoning

Gandalf is a **Wizard**

Knowledge and Reasoning

Knowledge

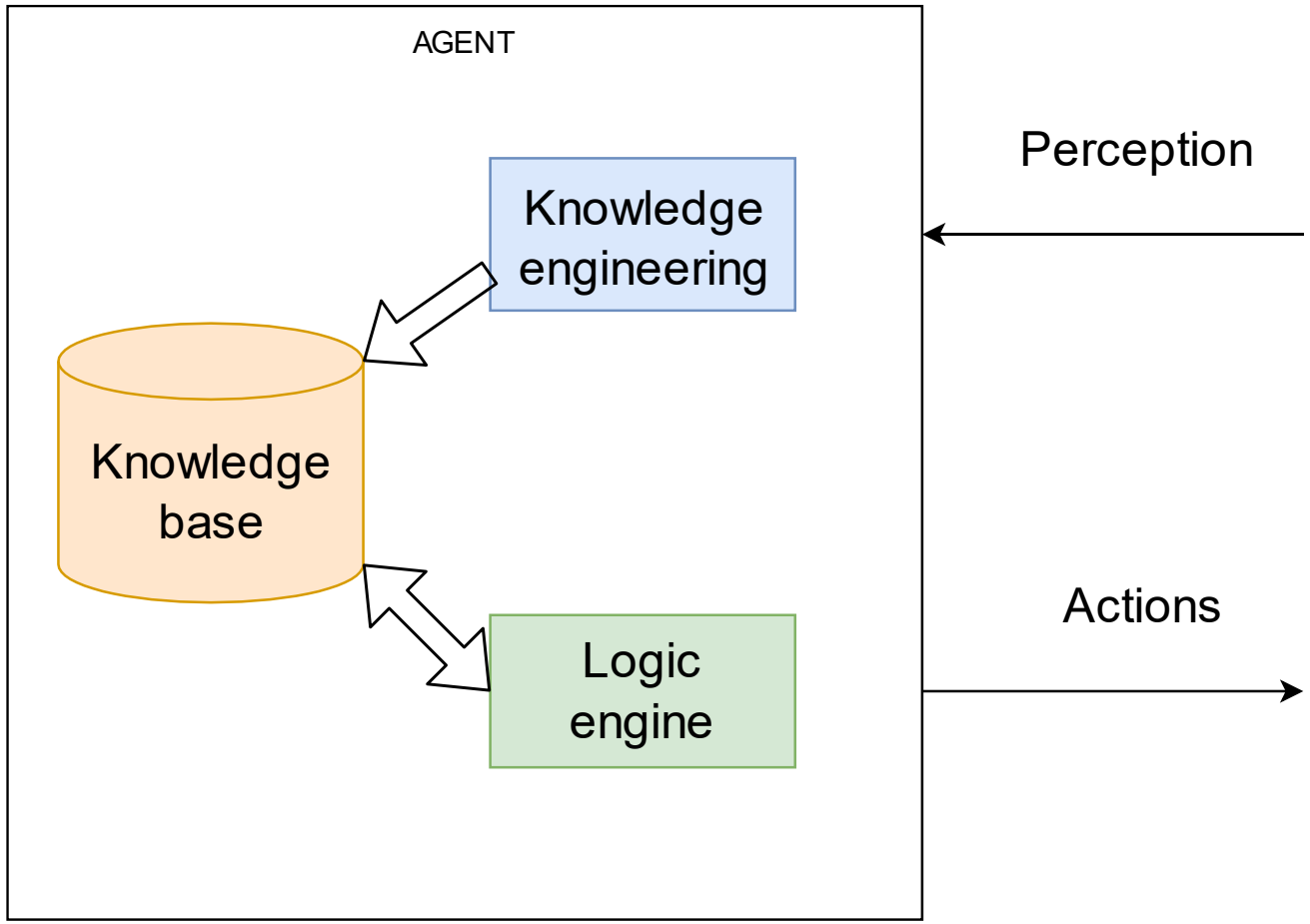
Wizards wear a wizard hat
Gandalf wears a wizard hat

Reasoning

Gandalf is a **Wizard**

Obviously useful for AI, but how to implement it?

Logical Agent



Knowledge Representation

Gandalf wears a wizard hat

Wizards wear a wizard hat

WizardHat(gandalf).

WizardHat(X) => Wizard(X).

Parsing the above by a computer:

Subject of decades of NLP research; still doesn’t work perfectly

Parsing is no more difficult than e.g. JavaScript

Wumpus World

A toy application domain

Wumpus World

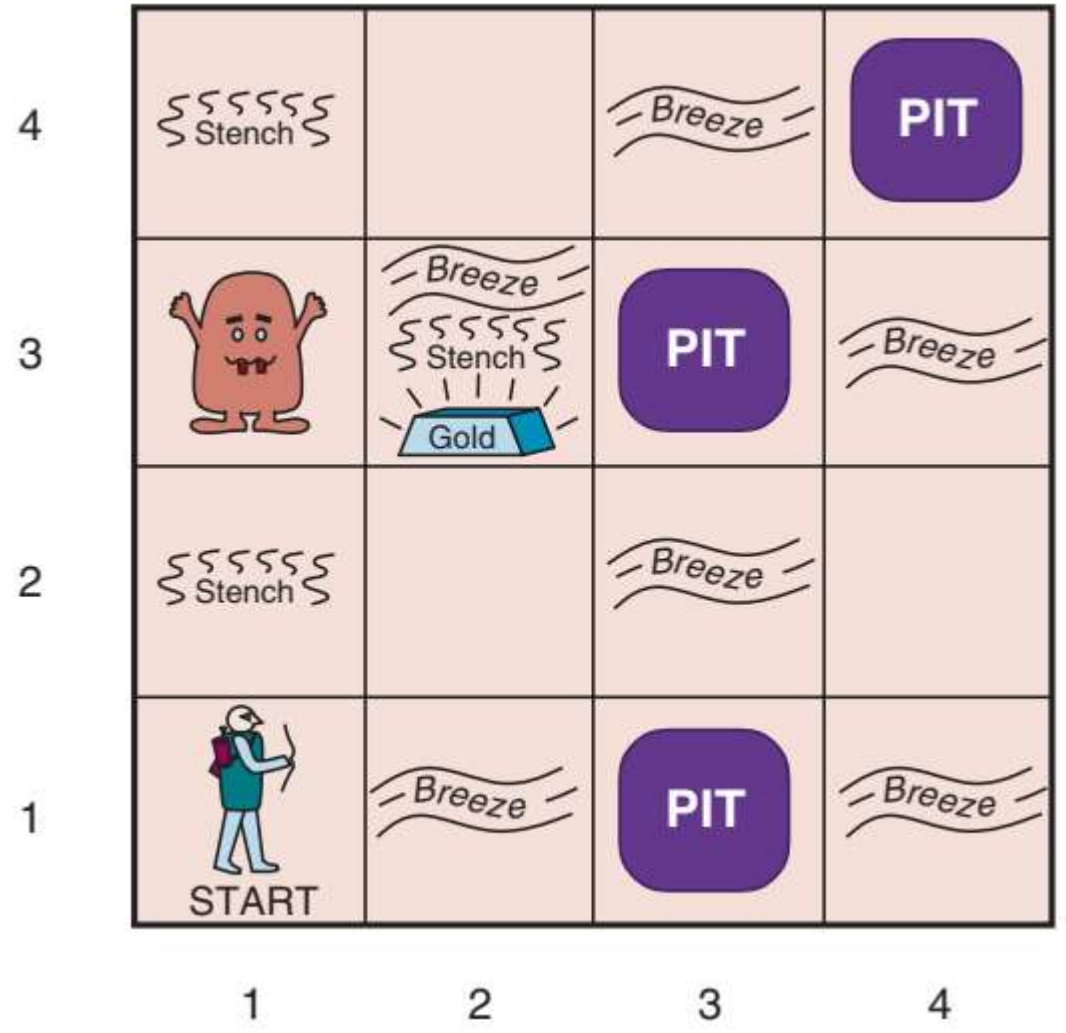
Adapted from 1970's computer game

Find gold in a 4x4 maze

Don't fall into a pit

Don't get eaten by the Wumpus

Sensor info: *stench* next to Wumpus, *breeze* next to pits



Reasoning in Wumpus World

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1

Reasoning in Wumpus World

1,4	2,4	3,4	4,4
1,3 **W!**	2,3	3,3	4,3
1,2 **A** **S** **OK**	2,2 **OK**	3,2	4,2
1,1 **V** **OK**	2,1 **B** **V** **OK**	3,1 **P!**	4,1

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

1,4	2,4 **P?**	3,4	4,4
1,3 **W!**	2,3 **A** **S G** **B**	3,3 **P?**	4,3
1,2 **S** **V** **OK**	2,2 **V** **OK**	3,2	4,2
1,1 **V** **OK**	2,1 **B** **V** **OK**	3,1 **P!**	4,1

Models and Worlds

The mechanism of machine reasoning

Example Reasoning Problem

Q: Is there a pit in [1,2], [2,2], [3,1]?

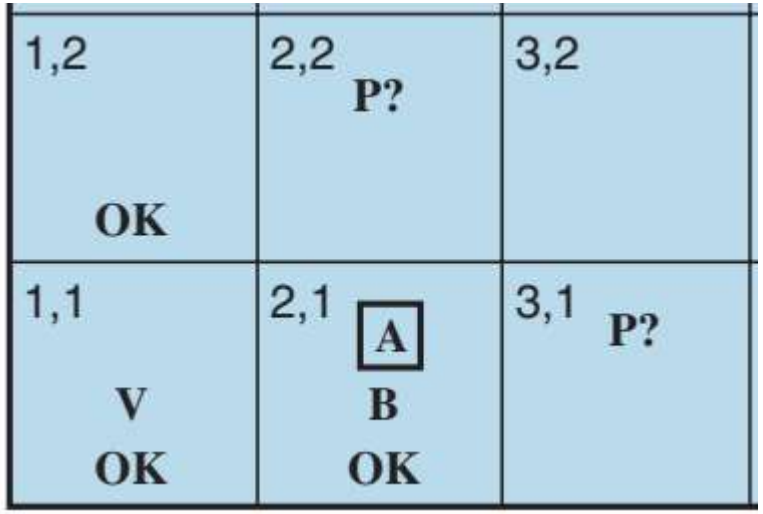
KB:

Breeze in [2,1]

No breeze in [1,1]

Rules about pits and breeze

(ignore stench/Wumpus for now)



Notation

α – some sentence (“there is no pit in [1,2]”)

m – one possible world

(pit in [2,2] and [3,1]; no pits in other squares)

m is a **model** of α, if α is true in m

KB knowledge base, a set of sentences

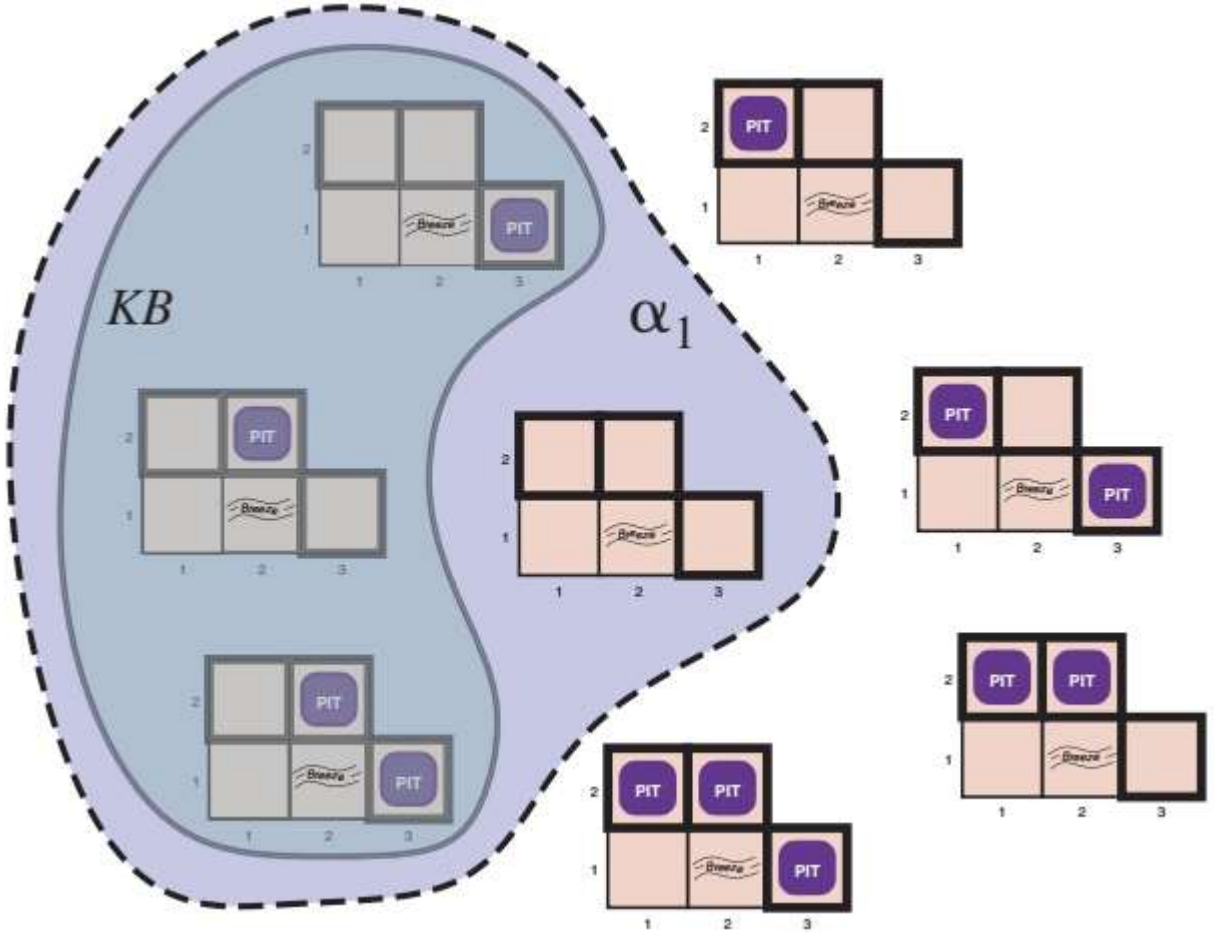
Entailment

α_1 – “there is no pit in [1,2]”

KB – “breeze in [2,1]; no breeze in [1,1]; rules”

KB restricts possible worlds

α_1 is true in all of them

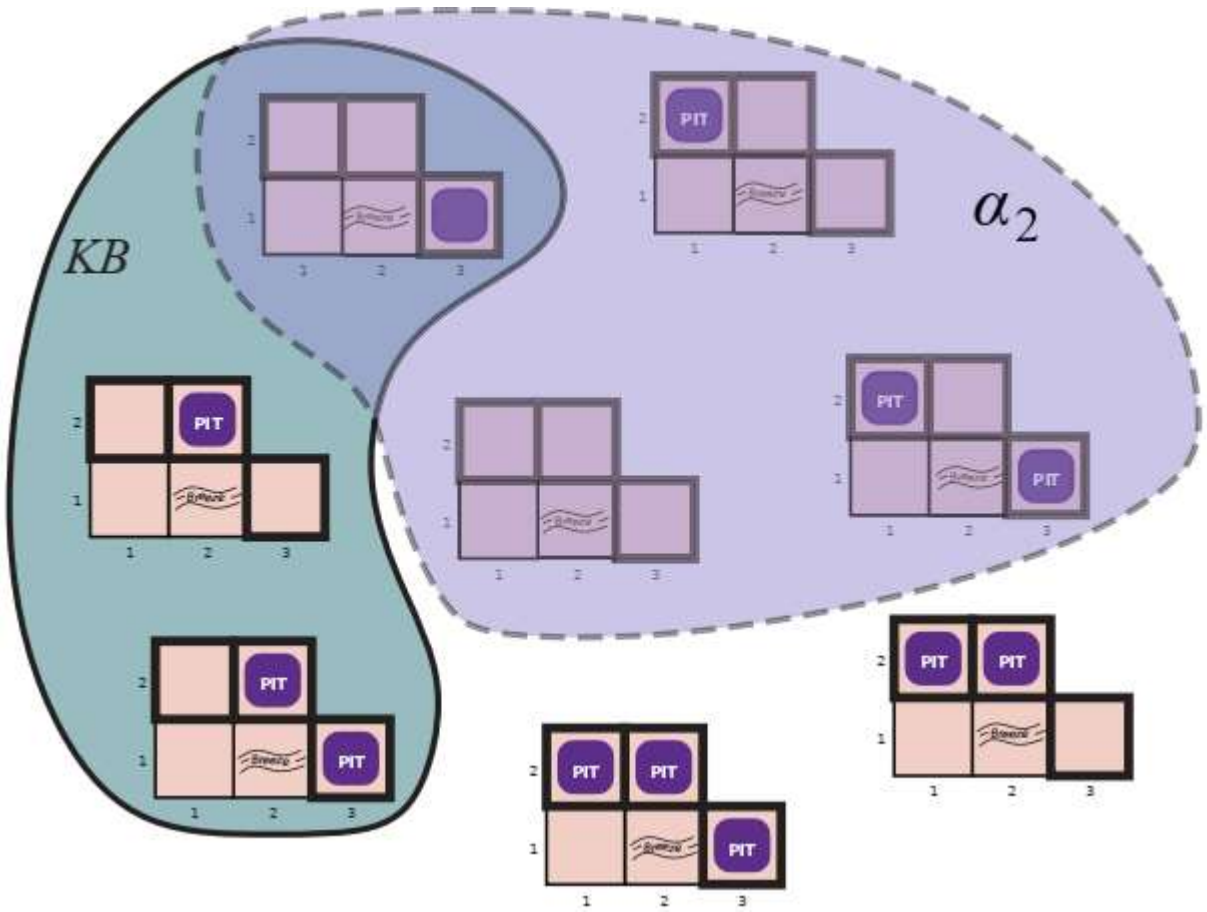


Entailment

α_2 – “there is no pit in [2,2]”

KB – “breeze in [2,1]; no breeze in [1,1]; rules”

α_2 is not true in some worlds that KB allows



Entailment

From previous examples

$$KB \models \alpha_1$$
$$KB \not\models \alpha_2$$

What we need:

1. Representation language, for KB-s and α-s
2. Inference procedures to discover entailments

   (right now we just looked at pictures; need actual algorithms)

Propositional Logic

The simplest form of logic

Logic

Provides:

- System to find truth values of sentences
- Inference rules; infer sentences from other sentences

Examples:

$(P \wedge Q) \vee Q$ is true, if Q is true

$\frac{P \rightarrow Q \quad P}{Q}$ We know P is true, Q always follows from P; Q must be true

Propositional Logic Syntax

P , $\neg P$ literal, negative literal.

α – a formula, examples:

P

$P \vee Q$

$Pit11 \rightarrow (Breeze12 \wedge Breeze21)$

Propositional Logic Semantics

Combine literals or sentences to new sentences using operators

α	$\neg\alpha$	$\neg\neg\alpha$
F	T	F
T	F	T

α	β	$\alpha \vee \beta$	$\alpha \wedge \beta$	$\alpha \rightarrow \beta$	$\alpha \Leftrightarrow \beta$
F	F	F	F	T	T
F	T	T	F	T	F
T	F	T	F	F	F
T	T	T	T	T	T

Each symbol and sentence has a truth value $(Breeze11 = false)$

Propositional Logic Inference

You can make your own rules, but here's a well known one:

Modus ponens

$$\frac{P \rightarrow Q \quad P}{Q}$$

Proof:

P	$P \rightarrow Q$	Q
F	T	F
F	T	T
T	F	F
T	T	T

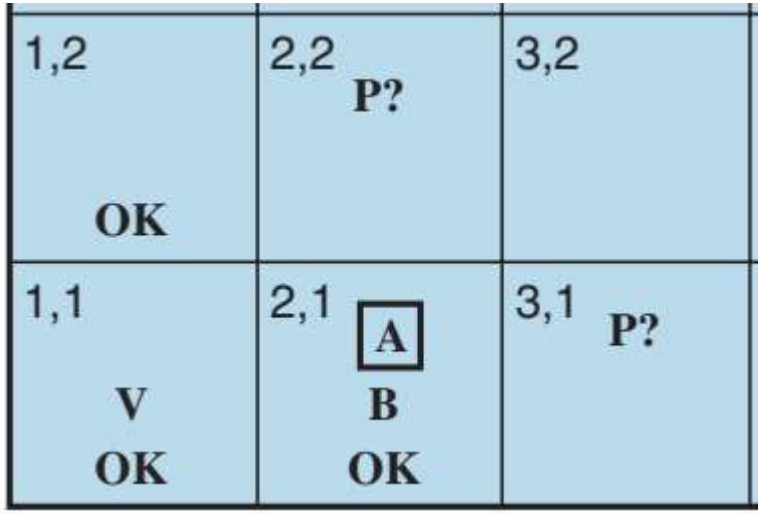
Model Checking

Our First Inference Method

Example KB

(Short notation: $B_{1,1}$ means “breeze in [1,1]”)

$R_1: \neg P_{1,1}$
$R_2: B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$
$R_3: B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$
$R_4: \neg B_{1,1}$
$R_5: B_{2,1}$



KB is our knowledge combined: $R_1 \wedge R_2 \wedge R_3 \wedge R_4 \wedge R_5$

Model Checking

Proving $KB \models \alpha$

Check ALL combinations of truth values for symbols
(all possible worlds)

Whenever KB is true, α must be true

Can implement with recursive DFS

Model Checking

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Model Checking

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮						⋮	⋮
false	*true*	*false*	*false*	*false*	*fal*						*ue*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Truth table shows that

$$KB \models \neg P_{1,2}$$

Model Checking

This algorithm is

sound – never derives a sentence that is not entailed

complete – always derives a sentence that is entailed

But very inefficient

$O(2^n)$ where n – number of symbols

Satisfiability

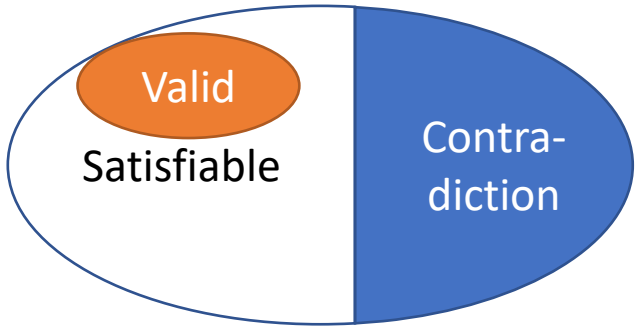
A problem related to model checking

Satisfiability

Sentences can be

valid; always true

$$B \vee C \vee \neg B$$



satisfiable:

true with some assignment of truth values

$$Cloudy \rightarrow Rains$$

unsatisfiable or contradictions; never true

$$A \wedge \neg A$$

Satisfiability

SAT – problem of finding if a logical formula is satisfiable

(a big deal in computer science)

Connection to entailment:

Deduction theorem $KB \models \alpha$ iff $KB \rightarrow \alpha$ is valid

$\equiv \neg(KB \rightarrow \alpha)$ is unsatisfiable

$\equiv \neg(\neg KB \vee \alpha)$ is unsatisfiable

$\equiv KB \wedge \neg\alpha$ is unsatisfiable

If the sentence is entailed, claiming the opposite should be a contradiction

Model checking revisited

More efficient model checking using satisfiability

- DPLL algorithm

  exploit logical structure to cut away some parts of the search tree

- WalkSAT: stochastic local search

  solution is one assignment of truth values; change values in search

  Not guaranteed to find a solution (**sound** but not **complete**)